PENJELASAN DARI KODE PEMBELAJARAN

PERTEMUAN 7

Nama : Muuhammad Luqmqnul Fikri

NIM : 231011400546

Kelas : 05TPLE017

□ Gambaran Umum

Script ini:

Membaca dataset processed_kelulusan.csv.

1. Membagi data menjadi train, validation, dan test set.
2. Menstandarkan fitur numerik.
3. Membangun model **ANN sederhana (fully-connected)**.
4. Melatih model dengan **early stopping** agar tidak overfitting.
5. Mengevaluasi performa model di test set.
6. Menyimpan model dan scaler.
7. Menampilkan hasil evaluasi (ROC, PR curve, dan learning curve).
8. Melakukan prediksi contoh satu data baru.

◈ 1. Inisialisasi dan Pengaturan Seed

```
# ---------- Seed utk reproducibility ----
SEED = 42
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)
```

- Menetapkan nilai *seed* agar hasil acak bisa **direproduksi** (hasil sama tiap dijalankan).
- Diterapkan ke random, numpy, dan TensorFlow.

🔷 2. Muat Dataset dan Split Data

```
df = pd.read_csv("processed_kelulusan.csv")

X = df.drop("Lulus", axis=1)
y = df["Lulus"]
```

- Membaca dataset kelulusan (harus sudah ada di folder).
- X = fitur prediktor, y = label biner (1 = Lulus, 0 = Tidak Lulus).

```
# 70/30 stratified
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=SEED
)
```

- 70% data untuk **train**, 30% sisanya untuk **validation + test**.
- stratify=y menjaga proporsi kelas 0/1 tetap seimbang.

Kemudian program mencari seed yang membuat test set berisi **dua kelas** (agar metrik seperti ROC bisa dihitung):

```
for rs in range(500):
    X_val_try, X_test_try, y_val_try, y_test_try = train_test_split(
```

- Setelah ketemu kombinasi yang pas, disimpan di seed_found.

🔷 3. Standardisasi Fitur

```
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_val_s   = scaler.transform(X_val)
X_test_s  = scaler.transform(X_test)
```

- Semua fitur numerik diskalakan menjadi distribusi standar (mean 0, std 1).
- Penting untuk ANN karena mempercepat konvergensi training.

🔷 4. Bangun Arsitektur ANN

Penjelasan:

```
model = keras.Sequential([
    layers.Input(shape=(X_train_s.shape[1],)),
    layers.Dense(32, activation="relu"),
    layers.Dropout(0.3),
    layers.Dense(16, activation="relu"),
    layers.Dense(1, activation="sigmoid")
])
```

- **Input layer**: jumlah neuron sesuai jumlah fitur.
- **Hidden layer 1**: 32 neuron, aktivasi ReLU.
- **Dropout (0.3)**: mencegah overfitting dengan mematikan 30% neuron secara acak saat training.
- **Hidden layer 2**: 16 neuron, ReLU.
- **Output layer**: 1 neuron sigmoid → keluaran probabilitas antara 0–1 (klasifikasi biner).

🔷 5. Kompilasi Model

```
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy", keras.metrics.AUC(name="AUC")]
)
```

- Optimizer: **Adam** (efisien dan stabil).
- Loss: **binary_crossentropy** (cocok untuk klasifikasi biner).
- Metrik tambahan: akurasi dan AUC.

🔷 6. Training dengan Early Stopping

```
# ---------- Langkah 3 — Training + EarlyStopping ----------
es = keras.callbacks.EarlyStopping(
    monitor="val_loss", patience=10, restore_best_weights=True
)
```

- Jika val_loss tidak membaik selama 10 epoch → training berhenti otomatis.
- Model kembali ke bobot terbaik (bukan epoch terakhir).

```
# dataset kecil → batch_size kecil juga
history = model.fit(
    X_train_s, y_train,
    validation_data=(X_val_s, y_val),
    epochs=200, batch_size=4,
    callbacks=[es], verbose=1
)
```

- Batch kecil (4) karena dataset kecil.
- Maksimal 200 epoch tapi berhenti lebih cepat jika val_loss stagnan.

🔷 7. Evaluasi Model di Test Set

```
# ---------- Langkah 4 — Evaluasi Test (acc/AUC Keras) --------
loss, acc, auc = model.evaluate(X_test_s, y_test, verbose=0)
```

- Mengukur *Loss*, *Accuracy*, dan *AUC* pada data test.

Prediksi probabilitas dan hasil klasifikasi (threshold default 0.5):

```
# Probabilitas & pred default threshold 0.5
y_proba_test = model.predict(X_test_s).ravel()
y_pred_test_050 = (y_proba_test >= 0.5).astype(int)
```

Kemudian tampilkan:

- Confusion matrix
- Classification report (precision, recall, F1)

🔷 8. Menentukan Threshold Optimal (berdasarkan Validation Set)

```
f1s = np.where((prec + rec) > 0, 2 * prec * rec / (prec + rec), 0)
best_idx = int(np.argmax(f1s))
best_thr = thr[best_idx] if best_idx < len(thr) else 0.5
print(f"\n[VAL] Best threshold by F1: {best_thr:.4f} | F1(val)~{f1s[best_idx]:.3f}")
```

- Menghitung nilai threshold terbaik berdasarkan **F1 tertinggi di validation**.
- Menggunakan threshold ini untuk prediksi ulang di test set.
- Ini membuat model lebih sensitif atau presisi tergantung data imbalance.

🔷 9. ROC dan Precision-Recall Curve

```
fpr, tpr, _ = roc_curve(y_test, y_proba_test)
plt.figure(); plt.plot(fpr, tpr, label="ROC (test)")
```

- Membuat **ROC curve** (TPR vs FPR) dan **PR curve** untuk mengukur keseimbangan antara precision dan recall.
- Disimpan sebagai file roc_ann_p7.png dan pr_ann_p7.png.

🔷 10. Learning Curve

```
# ---------- Langkah 5 — Visualisasi Learning Curve ----------
plt.figure()
plt.plot(history.history["loss"], label="Train Loss")
plt.plot(history.history["val_loss"], label="Val Loss")
plt.xlabel("Epoch"); plt.ylabel("Loss"); plt.legend()
plt.title("Learning Curve (ANN)")
plt.tight_layout(); plt.savefig("learning_curve_p7.png", dpi=120); plt.close()
print("Saved: learning_curve_p7.png")
```

- Grafik loss selama training → untuk melihat apakah model overfitting atau stabil.

🔹 11. Simpan Model dan Scaler

```
# ---------- Simpan Model + Scaler ----------
model.save("ann_p7.h5")
joblib.dump(scaler, "scaler_p7.pkl")
```

- Menyimpan model dalam format .h5 dan scaler dalam .pkl.
- Bisa dipakai untuk prediksi di waktu lain tanpa training ulang.

🔹 12. Demo Prediksi Data Baru

```
sample_s = scaler.transform(sample)
proba_sample = float(model.predict(sample_s).ravel()[0])
pred_sample  = int(proba_sample >= best_thr)
print(f"\nContoh prediksi sample → proba={proba_sample:.3f} | thr={best_thr:.3f} | pred={pred_sample}")
```

Membuat contoh input baru.

- Menstandarkan menggunakan scaler yang sama.
- Model memberi probabilitas lulus (proba_sample).
- Prediksi (pred_sample) ditentukan dengan threshold terbaik dari validation.

✅ Kesimpulan

Kode ini membangun sistem klasifikasi berbasis **ANN (Neural Network)** yang:

- **Ringan dan stabil** untuk dataset kecil.
- Memiliki **mekanisme otomatis** agar test memiliki dua kelas.
- Menggunakan **early stopping dan dropout** untuk mencegah overfitting.
- Mengoptimalkan threshold agar **F1 score maksimal**.
- Menghasilkan **grafik visualisasi dan model siap pakai**.

HASILNYA

```
[INFO] seed split kedua (val/test): 0
Shapes: (7, 5) (1, 5) (2, 5)
Label count — train:
 Lulus
1    4
0    3
Name: count, dtype: int64
Label count — val:
 Lulus
0    1
Name: count, dtype: int64
Label count — test:
 Lulus
1    1
0    1
Name: count, dtype: int64
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	192
dropout (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 16)	528
dense_2 (Dense)	(None, 1)	17

Total params: 737 (2.88 KB)

Trainable params: 737 (2.88 KB)

Non-trainable params: 0 (0.00 B)

V

```
Total params: 737 (2.88 KB)


Trainable params: 737 (2.88 KB)


Non-trainable params: 0 (0.00 B)


Epoch 1/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 4s 788ms/step - AUC: 0.2500 - accuracy: 0.2857 - loss: 0.7186 - val_AUC: 0.0000e+00 - val_accur
Epoch 2/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 181ms/step - AUC: 0.3333 - accuracy: 0.7143 - loss: 0.6975 - val_AUC: 0.0000e+00 - val_accur
Epoch 3/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step - AUC: 0.6667 - accuracy: 0.7143 - loss: 0.6648 - val_AUC: 0.0000e+00 - val_accur
Epoch 4/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step - AUC: 0.6667 - accuracy: 0.7143 - loss: 0.6579 - val_AUC: 0.0000e+00 - val_accur
Epoch 5/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 175ms/step - AUC: 0.5833 - accuracy: 0.4286 - loss: 0.6623 - val_AUC: 0.0000e+00 - val_accur
Epoch 6/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - AUC: 0.7083 - accuracy: 0.5714 - loss: 0.6555 - val_AUC: 0.0000e+00 - val_accur
Epoch 7/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 313ms/step - AUC: 0.7500 - accuracy: 0.5714 - loss: 0.6627 - val_AUC: 0.0000e+00 - val_accur
Epoch 8/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 255ms/step - AUC: 0.9167 - accuracy: 0.8571 - loss: 0.5984 - val_AUC: 0.0000e+00 - val_accur
Epoch 9/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step - AUC: 0.8333 - accuracy: 0.5714 - loss: 0.5963 - val_AUC: 0.0000e+00 - val_accur
Epoch 10/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 196ms/step - AUC: 0.7083 - accuracy: 0.7143 - loss: 0.5718 - val_AUC: 0.0000e+00 - val_accur
Epoch 11/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 165ms/step - AUC: 0.9167 - accuracy: 0.7143 - loss: 0.5605 - val_AUC: 0.0000e+00 - val_accur
Epoch 12/200
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 145ms/step - AUC: 0.8750 - accuracy: 0.8571 - loss: 0.5686 - val_AUC: 0.0000e+00 - val_accur
Epoch 13/200
```

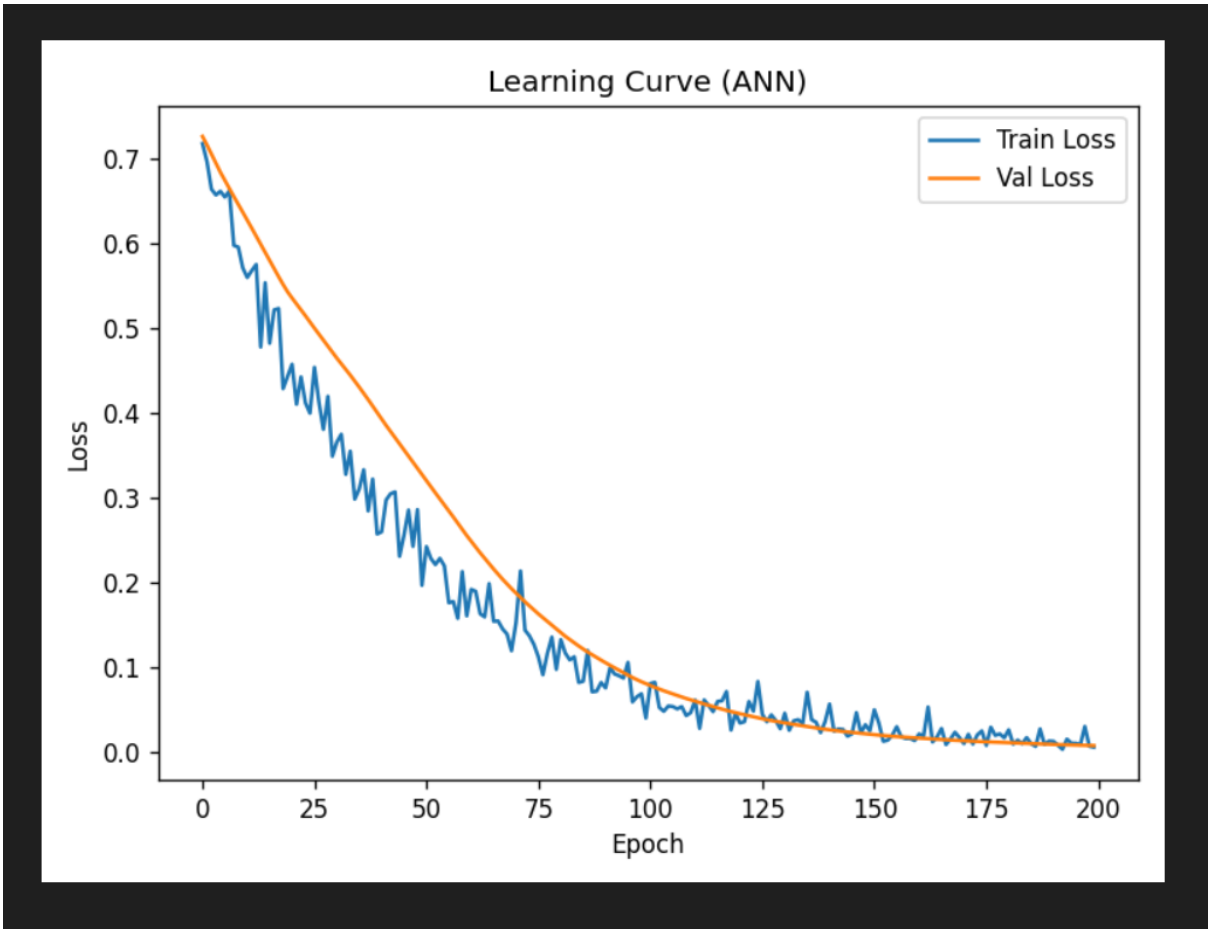
```
...
Confusion Matrix (test, best thr):
[[1 0]
 [0 1]]
ROC-AUC(test, sklearn): 1.0
```

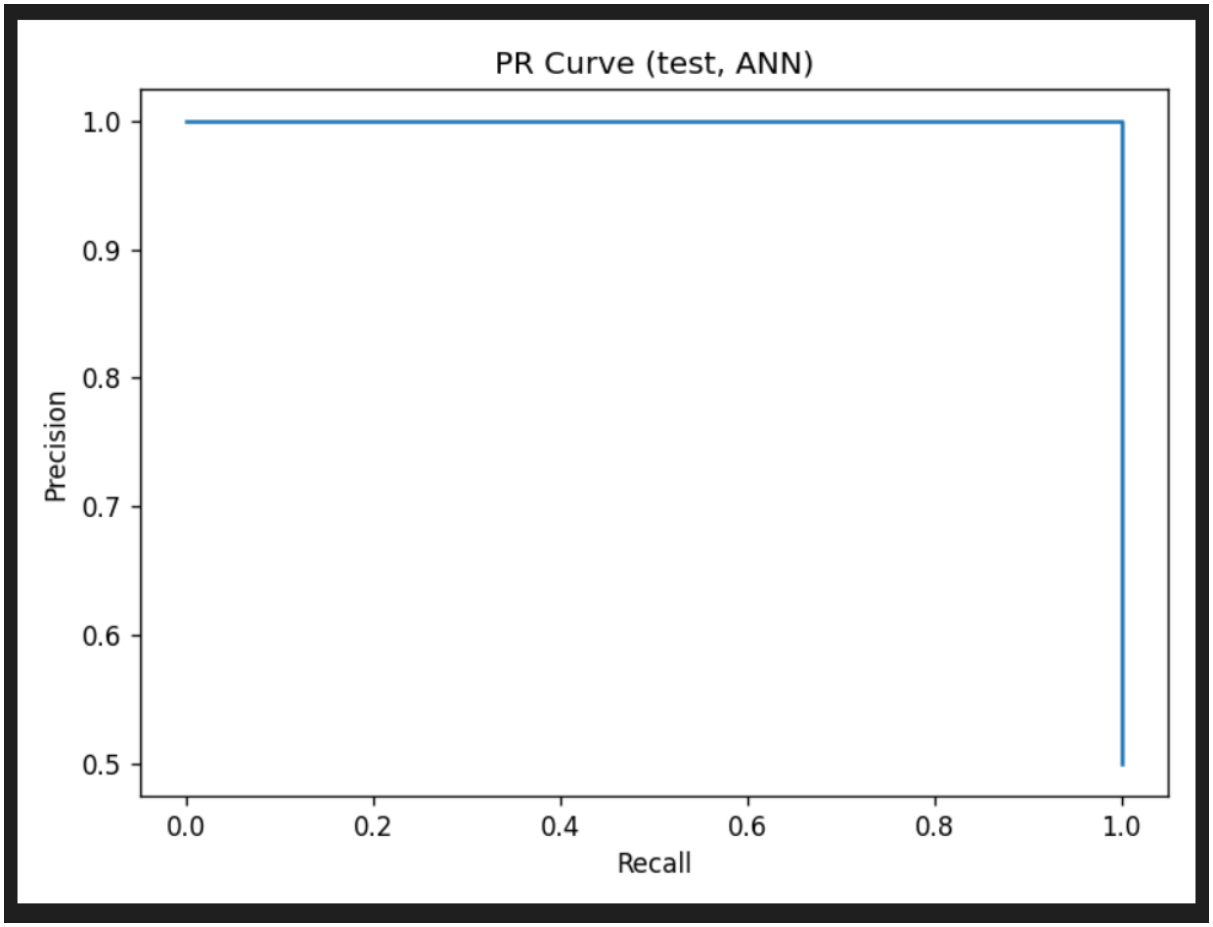
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

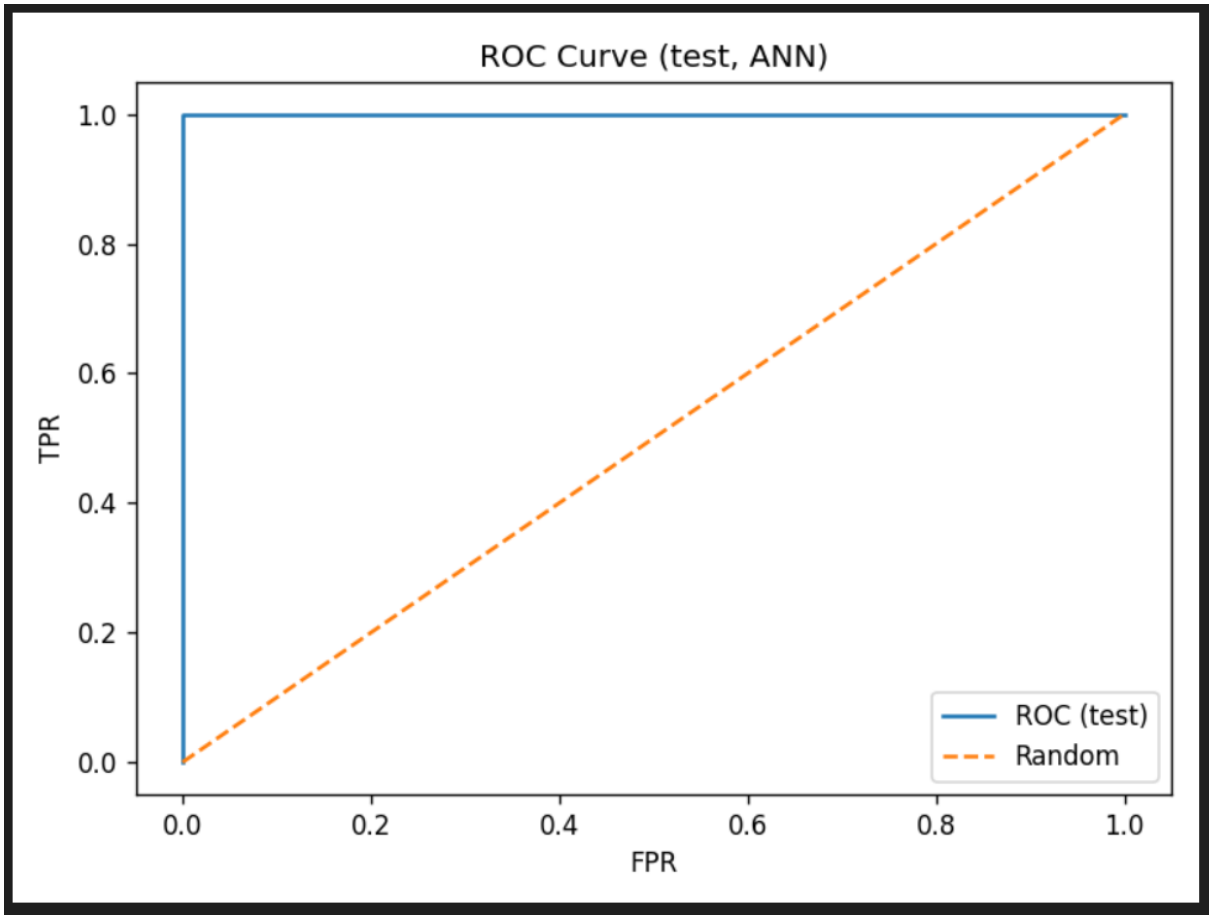
```
d:\machine_learning4\.venv\lib\site-packages\sklearn\metrics\_ranking.py:1046: UserWarning: No positive class found in y
  warnings.warn(
Saved: learning_curve_p7.png
Saved: roc_ann_p7.png
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file
Saved: pr_ann_p7.png

Saved model: ann_p7.h5 | scaler: scaler_p7.pkl
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 81ms/step

Contoh prediksi sample → proba=0.990 | thr=0.008 | pred=1
```


Learning Curve (ANN)
Train Loss
Val Loss
Loss
Epoch
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
25
50
75
100
125
150
175
200


PR Curve (test, ANN)
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve (test, ANN)
TPR
FPR
ROC (test)
Random
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0